# Conversión implícita

Se realiza de forma automática al asignar un valor de un tipo a otro compatible.

**Ejemplo:**

*byte x = 5;*

*int y = x;*

El valor almacenado en *x* (byte) se convertirá a entero al asignarse a *y* (int).

Las **posibles** conversiones implícitas son:

|  |  |
| --- | --- |
| **Nuevo tipo** | **Tipos origen** |
| short | byte |
| int | byte, short, char |
| long | byte, short, char, int |
| float | byte, short, char, int, long |
| double | byte, short, char, int, long, float |

# Conversión explícita

En este caso el programador especifica el nuevo tipo al que se va a transformar el dato. Se escribe de forma explícita entre paréntesis delante del dato.

**Ejemplo:**

*byte a = 20;*

*int x = (int) a;*

Al escribir int entre paréntesis se fuerza a cambiar el dato de tipo byte a int.

Hay que tener cuidado al realizar esta conversión ya que se puede aplicar a tipos no compatibles, lo que puede derivar en pérdidas de información e incluso errores en ejecución.

**Ejemplo:**

*float x = 5.7F;*

*int y = (int) x;*



¡¡¡ OJOOOOOOOOOOOOOOOOOOOO ¡¡

Se realizará la conversión, pero se perderá la parte decimal del número con punto flotante al guardarlo en *y*. Solo se guardará 5 en *y*.

Aunque parezca ilógico, habrá ocasiones en las que este tipo de conversiones nos pueden resultar útiles.

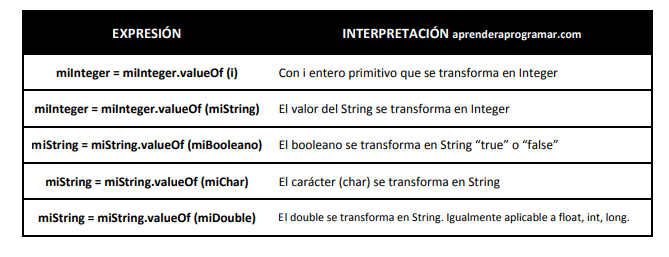
Por ejemplo, para extraer la parte entera de un número que tiene decimales.

Pero de forma general trataremos de atenernos a la norma de que "en las conversiones debe evitarse la pérdida de información".

## MÉTODO VALUEOF PARA CONVERSIÓN DE TIPOS

El método valueOf es un método sobrecargado aplicable a numerosas clases de Java y que permite realizar conversiones de tipos.

Veamos algunos ejemplos de uso.



Ejemplo:

**double miDouble=33.456;**

**String miCadena=miCadena.valueOf(miDouble);**

**// miCadena contiene ahora “33.456”, que es un String**